

Available online at www.sciencedirect.com



Journal of Computational Physics 200 (2004) 368-382

JOURNAL OF COMPUTATIONAL PHYSICS

www.elsevier.com/locate/jcp

# Local level set method in high dimension and codimension

Chohong Min \*,1

Department of Mathematics, University of California, Los Angeles, Los Angeles, CA 90095, USA

Received 24 November 2003; received in revised form 9 April 2004; accepted 13 April 2004 Available online 10 June 2004

#### Abstract

A new method is presented for numerically capturing a moving interface of arbitrary dimension and codimension. The method is named the '*local level set method*', since it localizes the *level set method* near the interface to significantly reduce the computational expense of the level set method. Following the framework of the level set method, an interface is implicitly represented as the zero level set of a vector valued function. A spatial tree structure is used to locally sample the vector valued function near the interface. Using a Lipschitz stable interpolation and a semi-Lagrangian scheme, our method is stable under both the maximum norm and the Lipschitz semi-norm. Due to this stability, the method does not need to reinitialize a level set function. Several numerical examples with high codimension are successfully tested. © 2004 Elsevier Inc. All rights reserved.

#### 1. Introduction

The *level set method* in [20] has been a successful tool for simulating a moving interface of codimension one, because of its simplicity and efficiency. Its successful applications include multiphase flows [17,19], surface reconstructions [6,7], and image processing [14,22]. One drawback of the level set method is its high computational expense because it expands the domain of computation from the interface to a grid in one higher dimension. To reduce the cost, two main approaches have been employed. One approach is to restrict the domain of a uniform grid near the interface [5,4]. The other approach uses a multi-resolution grid to enable high resolution only near the interface using a spatial tree structure, a so-called *quadtree* in  $\mathbb{R}^2$  and an *octree* in  $\mathbb{R}^3$  [11,15].

The level set method of codimension one [20] was theoretically extended to simulate a moving interface of arbitrary codimension [12], where an interface is implicitly represented as the zero level set of a *scalar* valued function, and applied, e.g., to a medical active contouring of codimension two [13]. However, the theoretical extension to higher codimension is unstable for locating isosurfaces. The instability was fixed in

<sup>\*</sup> Tel.: +1-310-486-4210.

E-mail address: chohong@math.ucla.edu (C. Min).

<sup>&</sup>lt;sup>1</sup> Research supported by ONR Grant N00014-03-1-0071 and NSF Grant DMS-0312222.

[18] by representing the isosurface as the zero level set of a *vector* valued function, and successfully applied to a moving curve in  $\mathbb{R}^3$  with geometry-dependent speed [18] and recently to capturing a one-dimensional wavefronts in  $\mathbb{R}^3$  and two-dimensional wavefronts in  $\mathbb{R}^5$  [21].

A drawback of the level set method in high codimension is its inefficiency by its expanding the domain of computation from the interface to a grid in higher dimension. It is the purpose of this paper to introduce a numerical method overcoming this drawback, while keeping the simplicity and versatility of the level set method. Our method originates from [11] in using a spatial tree structure and a semi-Lagrangian scheme, and is a generalization of [11], because it can deal with a moving interface of arbitrary codimension and does not require the reinitialization procedure every step.

The key concept of our method is to put more grid points near the interface and less grid points away from the interface, and implemented by using a tree structure that enables a multi-resolution grid. A grid cell keeps being split if it is near to the interface. The splitting condition can be easily formulated using the Lipschitz constant of the level set function and the magnitude of the function values, as shown in [11,24]. Another key concept is to make all the building blocks in our method Lipschitz stable, because the sampling process in our method heavily relies on the Lipschitz constant of a vector valued function.

A sampling algorithm is presented in Section 2 that adaptively samples a vector valued function. In Section 3, an interpolation algorithm reconstructs a Lipschitz continuous function from the sampled function. In Section 4, an evolution algorithm of a level set function is introduced that is Godunov-type, i.e., a combination of the sampling algorithm in Section 2, a approximate solution operator, and the interpolation algorithm in Section 3. A practical implementation of our method is given in Section 5. Our method is tested with several examples in Section 6.

#### 2. Adaptive sampling

Let a vector valued function  $\vec{\phi} : \mathbb{R}^d \to \mathbb{R}^c$  be given with its level set  $\Gamma \subset \mathbb{R}^d$ . Since the level set method eventually deals with  $\Gamma$ , not with  $\vec{\phi}$ , it is desirable to adopt a multi-resolution grid to enable a fine grid near  $\Gamma$  and a coarse grid away from  $\Gamma$ . For this purpose, we employ a spatial tree structure, called a Quadtree in  $\mathbb{R}^2$  and a Octree in  $\mathbb{R}^3$ , which has been a very successful tool in many areas in which multi-resolution is needed [8,9].

If the interface  $\Gamma(t) \subset \mathbb{R}^d$  moves with velocity  $\vec{V} : \mathbb{R}^d \to \mathbb{R}^d$ , the level set method implicitly tracks the interface as the zero level set of  $\vec{\phi}(t) : \mathbb{R}^d \to \mathbb{R}^c$  by solving a convection equation;

$$\vec{\phi}_t + (\vec{V} \cdot \nabla)\vec{\phi} = 0.$$

Since the solution of this equation stays Lipschitz continuous if the initial data is, it is valid to assume that  $\vec{\phi} : \mathbb{R}^d \to \mathbb{R}^c$  is Lipschitz continuous, that is

$$\operatorname{Lip}(\vec{\phi}) := \sup_{x \neq y \in \mathbb{R}^d} \frac{\|\vec{\phi}(x) - \vec{\phi}(y)\|_{\infty}}{\|x - y\|_2} < \infty.$$

Let a hierarchical grid  $\{G^l\}_{l=0,\dots,l_{\max}}$  be given in  $\mathbb{R}^d$  such that  $G^l$  has  $2^{l\cdot d}$  number of grid cells of equal size, and  $G^{l+1}$  is a refinement of  $G^l$ . Fig. 1 illustrates a nested grid in  $\mathbb{R}^2$ . Given a grid cell  $C \in G^l$ , let us denote its parent cell in  $G^{l-1}$  by prnt(C), the set of its children in  $G^{l+1}$  by children(C), its size by size(C), and the set of the center points of all the faces of C by centers(C). For a grid cell  $C = \prod_{i=1,\dots,d} [a_i, b_i]$ , centers(C) =  $\prod_i \{a_i, \frac{a_i+b_i}{2}, b_i\}$  and size(C) =  $\sqrt{\sum_i (b_i - a_i)^2}$ . Fig. 2 illustrates the set of center points of a unit cube  $[0, 1]^d$  when d = 1, 2, 3.

In the hierarchical grid  $\{G^{l}\}_{l=0,\dots,l_{\text{max}}}$ , we need to determine which cells are to be sampled. One rule of the determination is to guarantee that a parent cell is sampled whenever its child is sampled. This rule implies



Fig. 1. Nested grids in  $\mathbb{R}^2$ .



Fig. 2. Center points of a unit cube.

that there is always an hierarchical chain of grid cells from the root cell  $G^0$  to a sampled cell, and make the domain D form a tree structure, which allows fast access to its elements. The other rule is to ensure high resolution of sampling near the interface  $\Gamma$  and low resolution away from  $\Gamma$ . Satisfying these two rules, a very useful sampling algorithm, known as Whitney decompositions, appeared in [11] such that "recursively split and sample any cell whose edge length exceeds its minimum distance to  $\Gamma$ ".

For a general Lipschitz continuous function  $\vec{\phi} : \mathbb{R}^d \to \mathbb{R}^c$ , the Whitney decompositions can be generalized as

Recursively split and sample any cell satisfying  $\eta$ . The splitting condition  $\eta$  is given by

 $\eta(C): \min_{v \in \operatorname{centers}(C)} \left\| \vec{\phi}(v) \right\|_{\infty} \leq \operatorname{Lip}\left(\vec{\phi}\right) \cdot \frac{\operatorname{size}(C)}{4}.$ 

Algorithm 1 is a concrete formulation of our adaptive sampling algorithm. The algorithm starts with the root grid cell  $G^0$ , and recursively constructs a subdivision  $S = \{C_i\}$  of  $G^0$ .

Algorithm 1 (Subdivision of  $G^0$  according to  $\vec{\phi} : \mathbb{R}^d \to \mathbb{R}^c$ ). Input:  $G^0$  and  $\vec{\phi} : \mathbb{R}^d \to \mathbb{R}^c$ 1.  $C = G^0$  and  $S = \emptyset$ 2.  $S = S \cup \{C\}$ 3. if  $C \notin G^{I_{\max}}$  and  $\eta(C)$  is true 4.  $S = S - \{C\}$ 5. for all  $C' \in children(C)$ 6. go to 2 with C = C'Output: S From the subdivision  $S = \{C_i\}$  of the root cell  $G^0$ , a set  $D \subset \mathbb{R}^d$  is defined as

$$D = \bigcup_i \operatorname{centers}(C_i).$$

A sampled function  $\Re \vec{\phi}$  of  $\vec{\phi}$  is accordingly defined as a restriction of  $\vec{\phi} : \mathbb{R}^d \to \mathbb{R}^c$  to *D*, i.e.,  $\Re \vec{\phi} \doteq \vec{\phi} \mid_D : D \to \mathbb{R}^c$ . The following is a proposition explaining the properties of the sampling algorithm.

**Proposition 2.1.** Every grid cell intersecting  $\Gamma$  is contained in the subdivision S.

**Proof.** Given a grid cell C such that  $C \cap \Gamma \neq \emptyset$ , let  $w \in C \cap \Gamma$ . Then there exist a  $v \in \text{centers}(C)$  such that  $\|v - w\|_2 \leq \frac{\text{size}(C)}{4}$ , since the maximum distance in a cube is its diagonal size. Then,

$$\left\|\vec{\phi}(v)\right\|_{\infty} = \left\|\vec{\phi}(v) - \vec{\phi}(w)\right\|_{\infty} \leq \operatorname{Lip}(\vec{\phi}) \cdot \left\|v - w\right\|_{2} \leq \operatorname{Lip}(\vec{\phi}) \cdot \frac{\operatorname{size}(C)}{4}.$$

Therefore  $\eta(C)$  is true. Since  $C \subset \text{prnt}(C)$ ,  $\text{prnt}(C) \cap \Gamma \neq \emptyset$  and  $\eta(\text{prnt}(C))$  is true. Also every ancestor of *C* satisfies the splitting condition  $\eta$ . Since Algorithm 1 recursively constructs from an ancestor to its successor satisfying  $\eta$ , *C* is included in the subdivision *S*.  $\Box$ 

By the above proposition, the highest resolution is guaranteed near  $\Gamma$ . If every component  $\phi_i$  of  $\vec{\phi}$  is a signed distance function to its zero level set  $\{x \in \mathbb{R}^d | \phi_i(x) = 0\}$ , then the splitting condition is equivalent to a statement that the distance from C to  $\Gamma$  is smaller than one fourth of the size of C. Therefore, the memory size in a grid  $G^l$  is about the order of  $\Gamma$  which is (d-c) dimensional. Let  $N := 2^{l_{\text{max}}}$ , then  $|D \cap G^l| = O(N^{d-c})$  for  $0 \le l \le l_{\text{max}}$ . Combining all grids from  $l = 0, \ldots, l_{\text{max}}$ , the total memory size, |D| will be  $O(N^{d-c} \cdot \log(N))$ . Compared to the memory size  $O(N^d)$  of the uniform sampling, Algorithm 1 significantly saves memory, while maintaining its highest resolution near the interfaces by Proposition 2.1.

#### 3. Interpolation

In Section 2, we discussed a sampling procedure from continuous functions to discrete functions. Here, we discuss the reverse, an interpolation procedure from discrete to continuous. On a uniformly sampled function, piecewise polynomial interpolation has been one of the best ways to achieve both accuracy and efficiency. However its direct use on a multi-resolution grid would lead to Lipschitz instability. Fig. 3 shows a case where the piecewise multi-linear interpolation invokes a Lipschitz instability on a multi-resolution grid in  $\mathbb{R}^2$ .

In Section 4, we shall introduce a semi-Lagrangian scheme that is a combination of the adaptive sampling in Section 2, approximate solution operator, and the interpolation in this section. The whole scheme



Fig. 3. The piecewise multi-linear interpolation invokes discontinuity at P. The interpolation values at P from grid cells A and B are respectively 1 and 0.

needs to be Lipschitz stable. For this purpose, we present a Lipschitz stable interpolation on a multiresolution grid.

# 3.1. Triangulation

Let  $\vec{\psi}: D \to \mathbb{R}^c$  be an adaptively sampled function with a subdivision  $S = \{C_i\}$  of  $G^0$  generated by Algorithm 1. To enable a Lipschitz stable interpolation of  $\vec{\psi}$ , the subdivision S needs to be refined into a triangulation. For this purpose, we employ a well-known refining algorithm, *pulling* [3]. A pulling of a point P on a subdivision  $S = \{C_i\}$  of  $G^0$  results in a subdivision T of  $G^0$  that is obtained by modifying each element of S as follows:

 $\begin{cases} P \notin C_i, \text{ then } C_i \in T, \\ P \in C_i, \text{ then for every facet } F \text{ of } C_i \text{ not containing } P, \text{ conv}(P,F) \in T, \end{cases}$ 

Let us give the center points of every k-dimensional face of  $C_i$  the label  $k, 0 \le k \le d$ . By sequentially pulling all the center points in order of non-increasing label, the cubic subdivision  $\{C_i\}$  is refined to a triangulation,  $T = \{T_i\}$ , which is a general triangulation procedure known as a *complete barycentric subdivision* [3,16]. Fig. 4 shows a cases in  $\mathbb{R}^2$ . Since the domain D is the union of all center points of each  $C_i$ ,  $T = \{T_i\}$  is a triangulation of the root cell  $G^0$  with vertices in D.

#### 3.2. Simplicial interpolation

In Section 3.1, the domain of  $\vec{\psi} : D \to \mathbb{R}^c$  was triangulated into simplices  $\{T_j\}$ . Based on this triangulation, we define an interpolation, or a prolongation  $\mathscr{P}\vec{\psi} : \mathbb{R}^d \to \mathbb{R}^c$  as follows:

$$\mathscr{P}ec{\psi}(x) = egin{cases} \mathscr{P}ec{\psi}|_{T_j}(x), & x\in T_j, \ \mathscr{P}ec{\psi}(x^*), & x
eq G^0. \end{cases}$$

Here,  $x^*$  denotes the nearest-point projection of x into  $G^0$ , i.e., the nearest point in  $G^0$  to x, which is well defined since  $G^0$  is convex. Fig. 5 shows some cases of projections in  $\mathbb{R}^2$ .  $\mathscr{P}\vec{\psi}|_{T_j}$  is defined as the unique linear interpolant of  $\vec{\psi}$  on the simplex  $T_j$ . The interpolation  $\mathscr{P}$  is Lipschitz stable by the following proposition.

Here, we define the Lipschitz constant of a discrete function  $\vec{\psi}: D \to \mathbb{R}^c$  as

$$\operatorname{Lip}(\vec{\psi}) = \max_{x \neq y \in D} \frac{\left\| \vec{\psi}(x) - \vec{\psi}(y) \right\|_{\infty}}{\|x - y\|_2}.$$



Fig. 4. Triangulation of a cubic subdivision by pullings in  $\mathbb{R}^2$ .



Fig. 5. Projection of x outside into  $x^*$  inside the domain.

**Proposition 3.1.** The interpolation  $\mathscr{P}$  is Lipschitz stable, such that  $\operatorname{Lip}(\mathscr{P}\vec{\psi}) = \operatorname{Lip}(\vec{\psi})$ 

**Proof.** Since  $\mathscr{P}\vec{\psi}$  is a linear interpolant on each simplex  $T_j$ ,  $\operatorname{Lip}(\mathscr{P}\vec{\psi}|_{T_j}) = \operatorname{Lip}(\vec{\psi}|_{T_j}) \leq \operatorname{Lip}(\vec{\psi})$ . On  $T_j \cap T_{j'}$ ,  $\mathscr{P}\vec{\psi}$  has two possible definitions  $\mathscr{P}\vec{\psi}|_{T_j}$  and  $\mathscr{P}\vec{\psi}|_{T_{j'}}$ . Because  $\{T_j\}$  is a triangulation,  $T_j \cap T_{j'}$  is also a simplex. Since there exits only one linear interpolant on a simplex, the two definitions must be the same. Therefore  $\mathscr{P}\vec{\psi}$  is continuous on  $G^0$ . Since  $\operatorname{Lip}(\mathscr{P}\vec{\psi}|_{T_j}) \leq \operatorname{Lip}(\vec{\psi})$  and  $\mathscr{P}\vec{\psi}$  is continuous on  $G^0$ ,

 $\operatorname{Lip}(\mathscr{P}\vec{\psi}|_{G^0}) \leq \operatorname{Lip}(\vec{\psi}).$ 

For any  $x, y \in \mathbb{R}^d$ ,  $||x^* - y^*||_2 \leq ||x - y||_2$ . Therefore,

$$\left\|\mathscr{P}\vec{\psi}(x)-\mathscr{P}\vec{\psi}(y)\right\|_{\infty}=\left\|\mathscr{P}\vec{\psi}(x^*)-\mathscr{P}\vec{\psi}(y^*)\right\|_{\infty}\leqslant \operatorname{Lip}\left(\mathscr{P}\vec{\psi}\mid_{G^0}\right)\cdot\|x-y\|_2.$$

So,  $\operatorname{Lip}(\mathscr{P}\vec{\psi}) \leq \operatorname{Lip}(\vec{\psi})$ . Since the domain of  $\vec{\psi}$  is a subset of the domain of  $P\vec{\psi}$ ,  $\operatorname{Lip}(\mathscr{P}\vec{\psi}) = \operatorname{Lip}(\vec{\psi})$ .  $\Box$ 

#### 4. Evolution

In this section, we present an algorithm numerically capturing an interface in high dimension and codimension. Let us assume that an interface  $\Gamma \subset \mathbb{R}^d$  of codimension *c* is moving with velocity  $\vec{V} : \mathbb{R}^d \to \mathbb{R}^d$ . It can be implicitly represented as the zero level set of  $\vec{\phi}(x,t) : \mathbb{R}^d \times \mathbb{R}^+ \to \mathbb{R}^c$ , where each component of the  $\vec{\phi}$  satisfies the so-called level set equation

$$\frac{\partial}{\partial t}\phi_i + \vec{V}\cdot\nabla\phi_i = 0$$

for i = 1, ..., c. In a uniform grid, the level set equation is usually discretized with higher order accurate ENO or WENO schemes in space and the Runge–Kutta methods in time [1,23], but in a multi-resolution grid, neighboring points may not exist, which makes it hard to apply conventional finite difference schemes. Because of that, a semi-Lagrangian scheme has been used for discretizing the level set equation on a multi-resolution grid when the codimension is one [11]. We extend the techniques of [11] to higher dimension and codimension.

Given  $\vec{\phi}^n: D^n \to \mathbb{R}^c$ , the next level set function  $\vec{\phi}^{n+1}: D^{n+1} \to \mathbb{R}^c$  is defined as

$$ec{\phi}^{n+1} = (\mathscr{R} \circ \mathscr{S} \circ \mathscr{P}) ec{\phi}^n$$

Here,  $\mathscr{P}$  is the interpolation operator in Section 3,  $\mathscr{S}$  is an approximate solution operator, and  $\mathscr{R}$  is the adaptive sampling operator in Section 2. Via the characteristic curve of the level set equation,  $\mathscr{S}$  is equivalent to an ODE solver to the backward characteristic ODE,  $\dot{x} = -\vec{V}(x,t)$ . We choose the Euler scheme for an ODE solver,  $\mathscr{S}$ , then we have

$$(\mathscr{SP}\phi^n)(x) = (\mathscr{P}\phi^n)(x - \Delta t \vec{V}(x, t^n)).$$

To adaptively sample  $\vec{\phi}^{n+1}$  from  $\mathscr{GP}\vec{\phi}^n: \mathbb{R}^d \to \mathbb{R}^m$ , we need an estimate on the Lipschitz constant of  $\mathscr{GP}\vec{\phi}^n$ , which is given in the following proposition.

### **Proposition 4.1.**

$$\operatorname{Lip}(\mathscr{GP}\vec{\phi}^n) \leqslant \operatorname{Lip}(\vec{\phi}^n) \cdot \left(1 + \Delta t \cdot \sup_{x \in \mathbb{R}^d} \left\|\nabla \vec{V}(x, t^n)\right\|_2\right).$$

**Proof.** By the definitions of S and P,

$$\begin{split} \left\| \mathscr{GP}\vec{\phi}^{n}(x) - \mathscr{GP}\vec{\phi}^{n}(y) \right\|_{\infty} &= \left\| P\vec{\phi}^{n}\left(x - \Delta t\vec{V}(x,t^{n})\right) - P\vec{\phi}^{n}\left(y - \Delta t\vec{V}(y,t^{n})\right) \right\|_{\infty} \\ &\leq \operatorname{Lip}\left(P\vec{\phi}^{n}\right) \cdot \left\| x - y + \Delta t\left(\vec{V}(x,t^{n}) - \vec{V}(y,t^{n})\right) \right\|_{2} \\ &\leq \operatorname{Lip}\left(\vec{\phi}^{n}\right) \cdot \left( \|x - y\|_{2} + \Delta t \cdot \left\| \vec{V}(x,t^{n}) - \vec{V}(y,t^{n}) \right\|_{2} \right). \end{split}$$

Applying the mean value theorem to the vector valued function  $\vec{V}$ , we have

$$\left\|\vec{V}(x,t^n)-\vec{V}(y,t^n)\right\|_2 \leq \sup_{z\in\mathbb{R}^d} \left\|\nabla\vec{V}(z,t^n)\right\|_2 \cdot \|x-y\|_2.$$

Here,  $\nabla \vec{V}$  denotes the deformation matrix of  $\vec{V}$  and  $\|\nabla \vec{V}\|_2$  is the matrix 2-norm that is the maximal singular value of the matrix  $\nabla \vec{V}$ . The proposition follows from these two inequalities.  $\Box$ 

#### 5. Implementation

We discuss an implementation of our method that consists of an adaptive sampling in Section 2, a Lipschitz stable interpolation in Section 3, and an evolution algorithm in Section 4. Since the evolution is a combination of sampling and interpolation, it is enough to discuss implementations of sampling and interpolation.

#### 5.1. Implementation of sampling

Since the sampling Algorithm 1 recursively subdivides the coarsest cube  $G^0 \in \mathbb{R}^d$ , a  $2^d$  branched tree is a natural choice for the data structure of the sampling. Possible implementations of this tree structure are thoroughly discussed in [8,9]. Among them, *region-based trees* and *matrix-based trees* are the most appropriate for our purpose. The region-based tree is fast for interpolation but requires more memory for construction, while the matrix-based tree may be multiply defined, possibly  $2^d$  times, so it is hard to modify the tree, but, each point in matrix-based tree is singly defined, so it becomes easy to modify. Our algorithms, which consists of sampling and interpolation, do not need any modification, but only need new creations of

adaptively sampled functions. For these reasons, we take a region-based tree as a choice of the tree implementation.

Unlike the uniform sampling, we cannot predict the memory size of the adaptive sampling before doing it. Therefore, the programming languages with dynamic memory allocations are preferred, such as C++ or Java. We have chosen C++ for better performance. An implementation of region-based tree is given in the following, when the adaptive sampling is performed to a vector valued function  $\vec{\phi} : \mathbb{R}^d \to \mathbb{R}^c$  with d = 5 and c = 3.

```
struct Node {
    int il, i2, i3, i4, i5;
    int size;
    };
    struct Leaf : public Node {
    float vs1[3][3][3][3][3];
    float vs2[3][3][3][3][3];
    float vs3[3][3][3][3][3];
    };
    struct Branch : public Node {
    Node* children[2][2][2][2][2];
};
```

The quantity node represents a cube,  $\{x \in \mathbb{R}^d \mid i_j \leq x_j \leq i_j + \text{size}, j = 1, ..., d\}$ , and has two possible types, Leaf and Branch. If a node does need to be split, it takes a type of Branch that has  $2^d$  number of children. If not, it takes a type of Leaf, and its center points are sampled and stored. Since centers $(C) = [0, \frac{1}{2}, 1]^d$  for a cube  $C = [0, 1]^d$ , it would sample  $3^d \cdot c$  number of center points for each leaf.

#### 5.2. Barycentric interpolation on a cube

Any cube  $C \subset \mathbb{R}^d$  is affinely isomorphic to  $[-1, 1]^d$  under translations and scalings. So, we need only to discuss an interpolation on  $[-1, 1]^d$ , and the general case will follow by the affine isomorphism. Since centers $([-1, 1]^d) = \{-1, 0, 1\}^d$ , let us assume a function  $\vec{\psi} : \{-1, 0, 1\}^d \to \mathbb{R}^c$ .

Although the barycentric interpolation is defined as a piecewise linear interpolant on the triangulation of the domain in Section 3, we do not explicitly triangulate a cube, but employ the following very efficient algorithm.

Given  $x \in [-1, 1]^d$ , the coordinates of x are sorted with a permutation J of  $\{1, \ldots, d\}$  such that

 $1 \ge |x_{J(1)}| \ge \cdots \ge |x_{J(d)}| \ge 0.$ 

For a notational convenience, let us set  $x_{J(0)} = 1$  and  $x_{J(d+1)} = 0$  to have  $|x_{J(0)}| \ge \cdots \ge |x_{J(d+1)}|$ . Define  $P_0 = \vec{0}$  and

$$P_i = P_{i-1} + \operatorname{sgn}[x_{J(i)}]\vec{e}_{J(i)}$$
 for  $i = 1, \dots, d$ .

Here,  $\vec{e}_i$  denotes the canonical *i*th unit vector and sgn(x) refers to the signum of x, i.e., H(x) - H(-x) with the Heaviside function H. Then, we have the following barycentric decomposition of x:

$$x = \sum_{i=1}^{d} x_{i} \vec{e}_{i} = \sum_{i=1}^{d} x_{J(i)} \vec{e}_{J(i)} = \sum_{i=1}^{d} |x_{J(i)}| \cdot \operatorname{sgn}[x_{J(i)}] \vec{e}_{J(i)} = \sum_{i=1}^{d} |x_{J(i)}| \cdot [P_{i} - P_{i-1}] = \sum_{i=0}^{d} (|x_{J(i)}| - |x_{J(i+1)}|) P_{i}$$

Since the barycentric interpolant  $\mathscr{P}\vec{\psi}: [-1,1]^d \to \mathbb{R}^c$  is a linear interpolant on a simplex conv $[P_0,\ldots,P_d]$ , we have the following formula:

$$\mathscr{P}\vec{\psi}(x) = \sum_{i=0}^{d} [|x_{J(i)}| - |x_{J(i+1)}|]\vec{\psi}(P_i).$$

Note that  $\vec{\psi}(P_i)$  is well defined, since  $P_i \in \{-1, 0, 1\}^d$  for all  $i = 0, \dots, d$ .

#### 5.3. Barycentric interpolation on a multi-resolution grid

Let an adaptively sampled function  $\vec{\psi} : D \to \mathbb{R}^c$  with  $D = \bigcup_i \text{centers}(C_i)$  be given. This section discusses an implementation of its barycentric interpolation  $\mathscr{P}\vec{\psi} : \mathbb{R}^d \to \mathbb{R}^c$ . Since  $\mathscr{P}\vec{\psi}(x)$  for  $x \notin G^0$  is defined as  $\mathscr{P}\vec{\psi}(x^*)$  with  $x^* \in G^0$ , it is enough to discuss an interpolation procedure of  $\mathscr{P}\vec{\psi}(x)$  with  $x \in G^0$ .

 $\mathscr{P}\vec{\psi}: \mathbb{R}^d \to \mathbb{R}^c$  is defined as the piecewise linear interpolation on the triangulation of the domain D of  $\vec{\psi}$ . Since the triangulation is expensive to implement, it is desired to circumvent the triangulation as done in the previous section. The complete barycentric subdivision, which is the algorithm triangulating D, sequentially refines the subdivision  $\{C_i\}$  of  $G^0$  by pulling all the center points of  $\{C_i\}$  of label  $d, \ldots, 0$ . As a result, a cube may be subdivided by any neighborhood sharing a face of dimension  $0, \ldots, d-1$ . Since so many (2<sup>2d</sup> in the worst case) neighboring cubes are involved for the interpolation on a cube, the interpolation procedure becomes expensive if we access all the neighboring cubes.

If we stop the pullings of the complete barycentric subdivision at the label d-1, it may not be a triangulation in general and therefore invoke discontinuities, but this provides a very efficient algorithm to implement and does not degrade numerical results too much according to our experiments in Section 6. In Fig. 4, the fourth picture indicates the complete barycentric subdivision, and the third one indicates the partial barycentric subdivision that stops pullings at the label d-1. From these reasons, let us employ this partial barycentric subdivision to obtain the following efficient algorithm.

Given  $x \in G^0$ , it is not difficult to access the smallest cube *C* containing *x* in  $\{C_i\}$ , because the subdivision  $\{C_i\}$  forms a tree structure that enables a fast access. Let us denote by *P* the center of *C*, by *Q* the intersection of the boundary of *C* and the line connecting *P* and *x*, and by *C'* the neighborhood of *C* sharing the (d-1)-dimensional face with *C* and containing *Q*. Fig. 6 illustrates a case in  $\mathbb{R}^2$ . If *C* is smaller than *C'*, then *C* needs to be subdivided by some center points of *C'*. Otherwise, it would be enough to use the barycentric interpolation only on *C*, which is already discussed in the previous section. Here follows a concrete formulation of the interpolation procedure.

Let the cube *C* be given with coordinates;  $C = \{y \in \mathbb{R}^d | a_i \leq y_i \leq b_i\}$ . Its center point *P* is given by  $P_i = \frac{a_i + b_i}{2}$  for i = 1, ..., d. Let us define  $\lambda \in \mathbb{R}$  as

$$\lambda = \min_{1 \leq i \leq d} \frac{b_i - P_i}{|x_i - P_i|}$$

Then Q, the intersection point between  $\partial C$  and the line connecting P and x is given by

$$Q = P + \lambda(x - P).$$



Fig. 6. Weak barycentric interpolation on a dyadic grid.

376

Let C' be the smallest cube in  $\{C_i\}$  containing Q, then

$$\mathscr{P}\vec{\psi}(x) = \begin{cases} (\mathscr{P}\vec{\psi})|_C(x) & \text{if } \operatorname{size}(C) \leqslant \operatorname{size}(C'), \\ \frac{1}{\lambda}(\mathscr{P}\vec{\psi})|_C(P) + (1 - \frac{1}{\lambda})(\mathscr{P}\vec{\psi})|_{C'}(Q) & \text{if } \operatorname{size}(C) > \operatorname{size}(C'). \end{cases}$$

Barycentric interpolations on a cube,  $\mathscr{P}\vec{\psi}|_{C}$  and  $\mathscr{P}\vec{\psi}|_{C}$  were given in Section 5.2.

#### 6. Numerical examples

Every example was programmed in C++ and run on a PC with 2.2 GHz CPU and 2 GB memory. Since the adaptive sampling in Section 2 is uniform near the interface by Proposition 2.1, an isosurfacing algorithm [2] on a uniform grid was used to isosurface and visualize the level set of adaptively sampled functions.

#### 6.1. Accuracy test

Our method employed the piecewise linear interpolation in space and the Euler method in time, which would result in a first-order accurate method in a uniform grid. Since a multi-resolution grid is used for the space discretization in our method, the course parts of the grid negatively affects the fine parts, which will weaken the first-order accuracy. Here, we numerically test the accuracy of our method.

As a test example, a circle of center  $(-\frac{1}{3}, -\frac{1}{3})$  and of radius  $\frac{1}{3}$  is advected with a constant velocity  $\vec{V} = (1, 1)$  until T = 0.625. Since the deformation matrix  $\nabla \vec{V}$  is the zero matrix, the Lipschitz constant stayed the initial Lipschitz constant for the whole timesteps. Table 1 shows that the convergence rate of our method is about a half.

# 6.2. Wave reflections in $\mathbb{R}^2$

It is well known that a high frequency wave behaves like a particle, which can be nicely represented as the Eikonal equation,

$$u_t + c(x) \cdot \|\nabla u\| = 0.$$

However, the classical viscosity solution of this equation does not allow superpositions that are natural in the wave phenomenon. To overcome this deficiency, Osher et al. [21] substituted the Eikonal equation with its characteristic equation that captures the wave fronts in phase space, which is

$$\vec{\phi}_t + (\vec{V} \cdot \nabla)\vec{\phi} = 0,$$

with velocity

$$\vec{V}(x, y, \theta) = (\cos \theta, \sin \theta, 0).$$

Table 1 Accuracy test

Grid	$L^1$ error	Rate
128 <sup>2</sup>	0.0286	_
256 <sup>2</sup>	0.0197	0.54
512 <sup>2</sup>	0.0132	0.58
$1024^2$	0.00928	0.51

As a test example, we take an initial wave front as a circle of center at (0,0) and of a radius  $\frac{1}{2}$  that is implicitly represented as the zero level set of

$$\vec{\phi}_0(x, y, \theta) = (x - \frac{1}{2}\cos\theta, y - \frac{1}{2}\sin\theta).$$

We take a domain of  $[-1,1]^2 \times [-\pi,\pi]$  whose boundary behaves a mirror, or a reflector of the wave. To incorporate the reflection, we modify the interpolation procedure in Section 3. Given  $\vec{\phi}: D \to \mathbb{R}^2$  with  $(x,y,\theta) \notin [-1,1]^2 \times [-\pi,\pi]$ ,  $\mathscr{P}\vec{\phi}(x,y,z)$  is defined as  $\mathscr{P}\vec{\phi}(x^*,y^*,\theta^*)$  with  $(x^*,y^*,\theta^*) \in [-1,1]^2 \times [-\pi,\pi]$  such that

$$x^{*} = \begin{cases} 2-x, & x > 1, \\ -2-x, & x < -1, \\ x, & |x| \leqslant 1, \end{cases} \quad y^{*} = \begin{cases} 2-y, & y > 1, \\ -2-y, & y < -1, \\ y, & |y| \leqslant 1, \end{cases} \quad \theta^{*} = \begin{cases} -\theta, & |x| \leqslant 1 \text{ and } |y| > 1, \\ \pi-\theta, & |x| > 1 \text{ and } |y| \leqslant 1, \\ \pi+\theta, & |x| > 1 \text{ and } |y| > 1, \\ \theta, & |x| \leqslant 1 \text{ and } |y| \leqslant 1. \end{cases}$$

The timestep  $\Delta t$  was chosen as  $\Delta t = \frac{\sqrt{\Delta x^2 + \Delta y^2 + \Delta \theta^2}}{2}$ . The Lipschitz constants of  $\vec{\phi}(x, y, \theta, t^n)$  was increased by  $L^{n+1} = L^n \cdot (1 + \Delta t)$  with  $L^0 = \sqrt{5}/2$ . The evolution was simulated until T = 1.5 (see Fig. 7). The numerical cost of memory is  $O(N^2) \simeq O(N^2 \log N)$  as expected (see Table 2).

# 6.3. Wave reflections in $\mathbb{R}^3$

This example is an extension of the previous example to  $\mathbb{R}^3$ . Osher et al. [21] substituted the Eikonal equation with its characteristic equation that captures the wave fronts in phase space, which is

$$\vec{\phi}_t + (\vec{V} \cdot \nabla)\vec{\phi} = 0,$$

with velocity

$$\vec{V}(x, y, z, \theta, \psi) = (\cos \theta \sin \psi, \sin \theta \sin \psi, \cos \psi, 0, 0).$$

As a test example, we take an initial wave front as a sphere of center at (0,0,0) and of a radius  $\frac{1}{2}$  that is implicitly represented as the zero level set of

$$\vec{\phi}_0(x, y, z, \theta, \psi) = \left(x - \frac{1}{2}\cos\theta\sin\psi, y - \frac{1}{2}\sin\theta\sin\psi, z - \frac{1}{2}\cos\psi\right).$$

We take a domain of  $[-1,1]^3 \times [-\pi,\pi] \times [0,\pi]$  whose spatial boundary,  $\partial [-1,1]^3$  behaves a mirror, or a reflector of the wave. To incorporate the reflection, we modify the interpolation procedure in Section 3. Given  $\vec{\phi}: D \to \mathbb{R}^3$  with  $(x, y, z, \theta, \psi) \notin [-1,1]^3 \times [-\pi,\pi] \times [0,\pi]$ ,  $\mathscr{P}\vec{\phi}(x, y, z, \theta, \psi)$  is defined as  $\mathscr{P}\vec{\phi}(x^*, y^*, z^*, \theta^*, \psi^*)$  with  $(x^*, y^*, z^*, \theta^*, \psi^*) \in [-1,1]^3 \times [-\pi,\pi] \times [0,\pi]$  such that

$$a^{*} = \begin{cases} 2-a, & a > 1, \\ -2-a, & a < -1, \\ a, & |a| \le 1 \end{cases} \text{ for } a = x, y, z,$$
$$\theta^{*} = \begin{cases} -\theta, & |x| \le 1 \text{ and } |y| > 1, \\ \pi-\theta, & |x| > 1 \text{ and } |y| \le 1, \\ \pi+\theta, & |x| > 1 \text{ and } |y| \le 1, \\ \theta, & |x| \le 1 \text{ and } |y| \le 1, \end{cases} \psi^{*} = \begin{cases} \pi-\psi, & |z| > 1, \\ \psi, & |z| \le 1, \\ \psi, & |z| \le 1, \end{cases}$$

378



Fig. 7. Reflections in  $\mathbb{R}^2$  simulated on a 1024<sup>3</sup> grid.

Table 2 Reflections in	$\mathbb{R}^2$					
Grid	Local level set method			Level set method		
	Space (MB)	Rate	Time (s)	Rate	Space (MB)	Rate
128 <sup>3</sup>	61.4	_	250	_	33.6	_
256 <sup>3</sup>	138	2.2	1160	4.6	268	8.0
512 <sup>3</sup>	290	2.1	5045	4.3	2147	8.0
1024 <sup>3</sup>	594	2.0	21,068	4.2	17,179	8.0

Timestep  $\Delta t$  was chosen as  $\Delta t = 0.5 \cdot \sqrt{\Delta x^2 + \Delta y^2 + \Delta z^2 + \Delta \theta^2 + \Delta \psi^2}$ . The Lipschitz constants of  $\vec{\phi}(x, y, z, \theta, \psi, t^n)$  was increased by  $L^{n+1} = L^n \cdot (1 + \Delta t)$  with  $L^0 = \sqrt{5}/2$ . The evolution was simulated until T = 0.15 (see Fig. 8). Table 3 shows a significant saving of the memory.



Fig. 8. Reflections in  $\mathbb{R}^3$  simulated on a 32<sup>5</sup> grid until 1.0 s.

Reflections in $\mathbb{R}^3$										
Grid	Local level set method				Level set method					
	Space (MB)	Rate	Time (s)	Rate	Space (MB)	Rate				
8 <sup>5</sup>	3.0	_	3	_	0.4	_				
16 <sup>5</sup>	53	17.7	52	17.4	12.6	32.0				
325	284	5.4	240	4.0	403	32.0				
645	1092	4.2	4257	17.7	12,885	32.0				

# 7. Conclusion

Table 3

We have introduced a new method that can track an interface in arbitrary dimension and codimension. By localizing the level set method near the interface, our new method significantly reduced the high computational expense of the level set method, while keeping the simplicity and efficiency of the level set method.

Our method is stable under both the maximum norm and the Lipschitz semi-norm. Due to its Lipschitz stability, the method does not need any reinitialization of level set function. However, without reinitializations, the Lipschitz constant of a level set function might keep increasing by the estimate in Section 4, which makes the method a bit less efficient. Numerical results show that our method is a half order accurate.

In the future, the author expects to employ the reinitialization algorithms in [10,21] to the tree structure in our method to improve the memory efficiency, and intends to improve the accuracy of our method.

# Acknowledgements

I thank my advisor, Stanley Osher for his enormous encouragement and excellent guidance. I am also grateful to the referees of this paper for their careful reviews and wonderful suggestions.

## References

- A. Harten, B. Engquist, S. Osher, S. Chakravarthy, Uniformly high-order accurate essentially non-oscillatory schemes III, J. Comput. Phys. 71 (1987) 231–303.
- [2] C. Min, Simplicial isosurfacing in arbitrary dimension and codimension, J. Comput. Phys. 190 (2003) 295-310.
- [3] C.W. Lee, Subdivisions and triangulations of polytopes, in: Handbook of Discrete and Computational Geometry, CRC Press LLC, 1997.
- [4] D. Adalsteinsson, J. Sethian, A fast level set method for propagating interfaces, J. Comput. Phys. 118 (1995) 269–277.
- [5] D. Peng, B. Merriman, H. Zhao, S. Osher, M. Kang, A PDE based fast local level set method, J. Comput. Phys. 155 (1999) 410– 438.
- [6] H.K. Zhao, S. Osher, R. Fedkiw, Fast surface reconstruction using the level set method, in: 1st IEEE Workshop on Variational and Level Set Methods, 8th ICCV, Vancouver, 2001, pp. 194–202.
- [7] H.K. Zhao, S. Osher, B. Merriman, M. Kang, Implicit and nonparametric shape reconstruction from unorganized data using a variational level set method, Comput. Vis. Image Understand. 80 (2000) 295–314.
- [8] H. Samet, The design and analysis of spatial data structures, Addison-Wesley, 1990.
- [9] H. Samet, Applications of spatial data structures: computer graphics, image processing and GIS, Addison-Wesley, 1990.
- [10] J. Strain, Fast tree based redistancing for level set computations, J. Comput. Phys. 152 (1999) 648-666.
- [11] J. Strain, Tree methods for moving interfaces, J. Comput. Phys. 151 (1999) 616-648.
- [12] L. Ambrosio, M. Soner, Level set approach to mean curvature flow in arbitrary codimension, J. Diff. Geom. 43 (1996) 693-737.
- [13] L. Lorigo, O. Faugeras, W. Grimson, R. Keriven, R. Kikinis, A. Nabavi, C. Westin, Codimension-two geodesic active contours for the segmentation of tubular structures, in: Conference on Computer Vision and Pattern Recognition, IEEE 2000, pp. 444–451.
- [14] L. Rudin, S. Osher, E. Fatemi, Nonlinear total variation based noise removal algorithms, Physica D 60 (1992) 259-268.
- [15] M. Droske, B. Meyer, M. Rumpf, C. Schaller, An adaptive level set method for medical image segmentation, Lecture Notes in Computer Science IPMI (2001) 416–422.
- [16] M.M. Bayer, Barycentric subdivisions, Pacific J. Math. 135 (1988) 1-16.
- [17] M. Sussman, P. Smereka, S. Osher, A level set approach to computing solutions to incompressible two-phase flow, J. Compt. Phys. 114 (1994) 146–159.
- [18] P. Butchard, L.-T. Cheng, B. Merriman, S. Osher, Motion of curves in three spatial dimensions using a level set approach, J. Comput. Phys. 170 (1999) 720–741.
- [19] R. Fedkiw, T. Aslam, S. Xu, The ghost fluid method for deflagration and detonation discontinuities, J. Compt. Phys. 154 (1999) 393–427.
- [20] S. Osher, J. Sethian, Fronts propagating with curvature dependent speed: algorithms based on hamilton-jacobi formulations, J. Comput. Phys. 79 (1988) 12–49.
- [21] S. Osher, L.-T. Cheng, M. Kang, H. Shim, Y.-H. Tsai, Geometric optics in a phase space based level set and eulerian framework, J. Comput. Phys. 179 (2002) 622–648.

- [22] T. Chan, L. Vese, Active contours without edges, IEEE Trans. Image Process. 10 (2001) 266-277.
- [23] X.-D. Liu, S. Osher, T. Chan, Weighted essentially non-oscillatory schemes, J. Comput. Phys. 126 (1996) 202-212.
- [24] Y.-H. Tsai, L.-T. Cheng, S. Osher, P. Burchard, G. Sapiro, Visibility and its dynamics in a PDE based implicit framework, UCLA CAM Report 03-22.